

IN THE SPECIFICATION

Please replace the paragraph beginning at page 6, line 15 with the following paragraph:

A section of the current H.264 specification applicable to the present invention states: "If codIOffset is larger than or equal to codIRange, a value of 1 is assigned to binVal, no renormalization is carried out and CABAC decoding is terminated." When decoding ~~end_of_slice_flag~~ END-OF-SLICE-FLAG, the last bit inserted in register codIOffset is ~~rbsp_stop_one_bit~~ RBSP STOP ONE BIT. When decoding an I PCM mode macroblock, the next bit decoded after terminating CABAC decoding could be either the syntax element PCM ALIGNMENT ZERO BIT or a first bit of the syntax element PCM BYTE.

Please replace the paragraph beginning at page 7, line 17 with the following paragraph:

Referring to FIG. 2, a diagram of the encoder apparatus 100 is shown. The encoder 100 generally comprises a block (or circuit) 104, a block (or circuit) 106, a block (or circuit) 108, a block (or circuit) 110, a block (or circuit) 112, a block (or circuit) 114, a block (or circuit) 116, a block (or circuit) 118, and a block (or circuit) 120. The block 104 generally parses a video, audio and/or image data signal (e.g., IN). The block 106 generally provides subtraction of temporal and/or spatial and/or

inter-band prediction(s) to remove redundancy. The block 108 generally performs a transform and quantization process (if needed). The block 110 generally performs an inverse transform and inverse quantization and delay process (if needed). The block 112 generally performs a zig-zag scan (or other serialization) of two dimensional data and binarization for binary arithmetic encoding. The block 112 only performs the serialization functions if needed on a particular bitstream. The block 114 generally performs arithmetic entropy encoding (AC). The block 116 generally performs pulse code modulation (PCM) encoding. If the output data from the block 104 is already PCM data, the block 116 simply passes the PCM data to the block 118. The block 118 chooses either arithmetic entropy coded data or raw PCM data. The block 120 presents a compressed bitstream (e.g., COMP).

Please replace the paragraph beginning at page 8, line 18 with the following paragraph:

Referring to FIG. 3, a diagram of the decoder apparatus 102 is shown. For H.264/MPEG4-AVC the choice between arithmetic entropy coded (AC) and PCM data may be made on each macroblock of data. In general, the decoder 102 reverses the steps performed by the encoder 100. In particular, the decoder 102 generally comprises a block (or circuit) 130, a block (or circuit) 132, a block (or circuit) 134, a block (or circuit) 136, a block (or

circuit) 138, a block (or circuit) 140, a block (or circuit) 142, a block (or circuit) 144 and a block (or circuit) 146. The block 130 generally receives the compressed bitstream COMP (e.g., from the block 120). The block 132 generally parses the bitstream COMP by choosing arithmetic entropy decoding or ~~presents~~ presenting raw PCM data ~~of~~ for each block of data as signaled in the compressed bitstream COMP. The block 134 generally performs arithmetic entropy decoding (AC). The block 136 generally performs pulse code demodulation. If the data signal is to remain as PCM data, the block 136 may simply pass the data to the block 146. The block 138 generally performs inverse binarization for binary arithmetic decoding and inverse zig-zag scan or other two-dimensionalization of the serial data. The block 138 may be an optional block. The block 140 generally performs inverse transform and quantization (if needed). The block 142 generally provides the delay of the output of the block 140 to present a separate input to the block 144. The block 144 generally performs ~~additional~~ addition of temporal and/or spatial and/or inter-band prediction(s) to restore redundancy. The block 146 generally combines the signals generated by the block 136 and the block 144 to present a video/audio and/or image data signal (e.g., OUT).

Please replace the paragraph beginning at page 11, line 11 with the following paragraph:

Referring to FIG. 4, a simplified block diagram of a binary arithmetic coding engine 150 is shown. The engine 150 generally comprises a block 152, a block 154, a block 156, a block 158, a block 160 and a block 162. The block 152 generally receives the value BINVAL (which is the current binary input symbol from the binarization block 112 of FIG. 2). The block 152 may also generate a context adaptive binary arithmetic encoder (CABAC) derived input (e.g., CTXIDX). The derived input CTXIDX may be a current context, which is derived from the history of the bitstream (e.g., from other symbols that have passed through the AC encoder previously).

Please replace the paragraph beginning at page 12, line 13 with the following paragraph:

The block 160 may generate the internal state variables in the registers of ~~the~~ the AC decoder (e.g., CODIOFFSET and CODIRANGE). The block 160 may also execute a procedure to modify the state of the registers and to renormalize the contents of the registers when necessary. The block 162 may generate the value BINVAL as the current binary output symbol to the inverse binarization block 138 of FIG. 3.

Please replace the paragraph beginning at page 13, line 4 with the following paragraph:

Referring to FIG. 5, a flowchart of a method (or process) 200 is shown. The method 200 generally comprises decoding a decision before CABAC termination that may be used in the arithmetic coding/decoding engines of FIGS. 2 and 3. The method 200 generally comprises a state 202, a state 204, a decision state 206, a state 208, a state 210, a state 212, and a state 214. The state 202 generally performs a decode terminate function. The state 204 generally decrements the first code range (e.g., CODIRANGE) by a value of two. The decision state 206 determines whether an offset (e.g., CODIOFFSET) is greater than or equal to the code range CODIRANGE. If so, the method 200 moves to the state 208 where the value BINVAL is set to 1. The method 200 then moves to the done state 214. If the decision state 206 determines that the code offset CODIOFFSET is not greater than or equal to the code range CODIRANGE, the method 200 moves to the state 210. The state 210 sets the value BINVAL equal to ± 0 and the method 200 moves to the state 212. The state 212 provides a renormalization. The method 200 then moves to the done state 214.

Please replace the paragraph beginning at page 14, line 1 with the following paragraph:

The method 200 may be invoked when encoding/decoding the END-OF-SLICE-FLAG or the BIN-INDICATING-I_PCM mode. The method 200 generally illustrates arithmetic decoding, which is the

standardized portion of the codec for H.264/MPEG4-AVC. In the state 206, an internal state register CODIRANGE is first decremented by 2, then compared with the register CODIOFFSET. Depending on the result of the comparison the binary symbol value BINVAL is output with either a value 0 (after which renormalization of the internal state of the arithmetic coding engine must occur) or a value 1. Note that it is in the method of operation of the decode terminate process that the current invention differs from conventional approaches: namely prior art permitted only the 'NO' branch of decision block 206 to be taken for decoding of the BIN-INDICATING-I_PCM mode. In contrast, the conventional approach states that if the ~~office offset~~ value CODIOFFSET is larger than or equal to the range value CODIRANGE, a value of 1 is assigned to BINVAL, no renormalization is carried out and CABAC decoding is terminated. In conventional approaches, the last bit inserted in the register CODIOFFSET is RBSP_STOP_ONE_BIT. In the new invention, other values (bit patterns) may be inserted as the last bit in the register CODIOFFSET, without impacting conformance to the H.264 standard. The effect of the additional bit patterns is that correct termination (e.g., decoding of the BINVAL symbol and renormalization of the internal state registers of the AC engine) may now be accomplished when decoding the BIN-INDICATING-I_PCM mode.